Group 9

Britt Binler, Jeremy Danziger, Wei-Yin Ko

I. Technical Documentation

1. What is the structure of the messages that are sent from the user interface to the middleware?
2. What is the structure of the messages that are sent from the middleware to the user interface?
3. What is the structure of the messages that are sent from the middleware to the sensor/display?
4. What is the structure of the messages that are sent from the sensor/display to the middleware?
5. How did you keep track of the average temperature?
6. What are the three additional features that you implemented?

II. User Manual

1. Convert temperature units displayed
2. Display local weather data
3. Display most recent temperature sensor reading

For the past hour (if the sensor has been running for less than an hour, the user should be able to see the statistics for the time since the sensor started running).

4. Display the average, low, and high temperature
5. Standby
6. Troubleshooting

III. Supplemental Information

1. Where does data processing occur?
2. What is the structure of your communication protocols between the different components?
3. What sorts of error-handling have you included in your code?
4. How are API calls made?

I. Technical Documentation

1. What is the structure of the messages that are sent from the user interface to the middleware?

In **`app.js`**, four different methods are used for our POST requests:

```
19 function checkPayload(e)
30 function sendStandBy()
134 function convertTemp()
153 function sendToServer()
```

In **`checkPayload(e)`**, the key variable refers to the state of the Pebble.

```
function checkPayload(e){
    for (var key in e.payload) break;
    console.log(JSON.stringify(e));

    if(key === '0'){
    checkConnect();
    setTimeout(getCity, 1500);
    setTimeout(sendToServer, 3000);
    } else if(key === '1'){
    checkConnect();
    setTimeout(sendStandBy, 1500);
    } else if (key === '2'){
    checkConnect();
    setTimeout(getFromServer, 1500);
    } else if (key === '3'){
    checkConnect();
    setTimeout(convertTemp, 1500);
    }
}
```

The **`key`** is set in main.c using the the button handler methods:

```
7  void down_click_handler(ClickRecognizerRef recognizer, void *context)
58 void select_click_handler(ClickRecognizerRef recognizer, void *context)
19 void double_click_handler(ClickRecognizerRef recognizer, void *context)
```

Back in **`app.js`**, the **`sendStandby()`** method sends POST request as:

```
123    var standby = 'STANDBY';
```

the `sendToServer()` method sends POST requests as:

```
var tempMsg = 'OUTSIDE TEMP '+ strTemp;
```

And the `convertTemp()` method sends POST requests as:

```
var convert = 'CONVERT';
```

2. What is the structure of the messages that are sent from the middleware to the user interface?

The middleware sends JSONs to the user interface, so that it may be parsed by the javascript.

Example:

```
{"temp": "24.250000C",
 "high": "24.312500C",
 "average": "24.281250C",
 "low": "24.250000C",
 "lastmotion": "3",
 "arduino": "yes"  }
```

```
Structure:
{"temp": [last read temperature][unit],
 "high": [high from last hour][unit],
 "average": [avg from last hour][unit],
 "low": [low from last hour][unit],
 "lastmotion": [# sec since last motion],
 "arduino": [yes/no status of arduino]}
```

3. What is the structure of the messages that are sent from the middleware to the sensor/display?

The middleware sends single characters to indicate to the Arduino that some action needs to be performed

Examples:
`"M"` - go into/out of standby
`"T"` - change the unit type you are in (convert between celcius/farenheit)
`"H"`, `"C"`, `"N"` - change the color of your LED based on the outside temperature (H for hotter, C for colder, N for nominal difference)

4. What is the structure of the messages that are sent from the sensor/display to the middleware?

The sensor sends the server a string to be parsed.

```
void SerialMonitorPrint (byte Temperature_H, int Decimal, bool IsPositive, int last_movement)
{
    Serial.print("\nS:");
    Serial.print(state);
    Serial.print(";T:");
    if (!IsPositive)
    {
      Serial.print("-");
    }
    Serial.print(Temperature_H, DEC);
    Serial.print(".");
    Serial.print(Decimal, DEC);
    Serial.print(";M:");
    Serial.print(last_movement);
    Serial.print(";\n");
}
```

From Arduino to Server: `"S:[F/C];T:___;M___;\n"`
`"T"` - temperature
`"M"` - time of last motion
`"S:F/C"` - State: Fahrenheit or Celsius

Example: `"S:F;T:77.3767;M:4;"`

5. How did you keep track of the average temperature?

In the **`linkedlist.h`** file, a linked list has been implemented such that it stores both the Celcius and Fahrenheit values of each reading. When **`get_average`** is called, the average temperature is calculated.

```
typedef struct temp_node {
  int addtime;  //the time it was added
  double temp_c;  //the temperature in celsius
    double temp_f;  //the temperature in celsius
  struct temp_node *next;  // pointer to next node
} node ;
```

```
16 double get_average(node* head_node);
```

A new node is added to the linked list each time the USB is read. The average temperature is calculated using **`get_average(head_node)`**.

```
120 double get_average(node* head_node) {
121     //return null for an empty list
122     if (head_node == NULL) {
123         return -1000;
124     }
125     //set up variables
126     node* current = head_node;
127     char current_top[150];
128     double total = 0;
129     int count = 0;
130 
131     //loop through 10 times, each time finding a new max (that is, the max not already
132     //in the top_ten list)
133     current = head_node;
134     while (current) {
135         total += current->temp;
136         count++;
137         current = current->next;
138     }
139     return total/count;
140 }
```

The method traverses the linked list, sums up all of the temperatures stored in each node while incrementing the count variable. Finally, the method will calculate the average by dividing the total sum by the node count.

6. What are the three additional features that you implemented?

indicate which parts of your code (including line numbers) implement these features

1. Based on the weather outside (retrieved from weather API), the color of the Arduino light will be either green (same as inside), red (warmer than inside), or blue (colder than inside).
    a. **`app.js`**:
        i.
        ```
        10 function getCity()
        180   req.open('GET', 'http://api.openweathermap.org/data/2.5/weather?'
        +
        ```

```
                'lat=' + 39.98 + '&lon=' + -75.19  + '&cnt=1&appid=' + key,
true);
```

b. **`tempandmotion.ino`**:
   i. `252 void Cal_temp (int& Decimal, byte& High, byte& Low, bool& sign)`
   ii. `380 void UpdateRGB (byte Temperature_H)`

```
161     /* Calculate temperature */
162     Cal_temp (Decimal, Temperature_H, Temperature_L, IsPositive);

168     /* Update RGB LED.*/
169     UpdateRGB (Temperature_H);
```

2. When the motion sensor detections a motion, the displayed temperature will be converted from Celsius to Fahrenheit and displayed on both Arduino and Pebble. The Pebble will also display time of last motion.
   a. **`tempandmotion.ino`**:

```
161     /* Calculate temperature */
162     Cal_temp (Decimal, Temperature_H, Temperature_L, IsPositive);
163
164     /* Display temperature on the serial monitor.
165       Comment out this line if you don't use serial monitor.*/
166     SerialMonitorPrint (Temperature_H, Decimal, IsPositive,(millis() -
last_movement)/1000);
167
168     /* Update RGB LED.*/
169     UpdateRGB (Temperature_H);
170
171     /* Display temperature on the 7-Segment */
172     if (mode == ACTIVE) {
173       Dis_7SEG (Decimal, Temperature_H, Temperature_L, IsPositive);
174     }

176     if(digitalRead(pirPin) == HIGH){
177        //digitalWrite(ledPin, HIGH);   //the led visualizes the sensors output
pin state
178        //digitalWrite(3, HIGH);
179        //digitalWrite(5, HIGH);
180        if(lockLow){
181         if (state == 'C') {
182           state = 'F';
183          } else {
184           state = 'C';
185          }
186          //makes sure we wait for a transition to LOW before any further
output is made:
187          lockLow = false;
188          //Serial.println("---");
189          //Serial.print("motion detected at ");
```

```
        last_movement = millis();
        //Serial.print(last_movement/1000);
        //Serial.println(" sec");
        delay(50);
        }
        takeLowTime = true;

      }
```

b. **server.c**:

c. **main.c**:

   i.
```
void in_received_handler(DictionaryIterator *received, void *context)
```

d.

3. The Pebble will show a weather description of the weather outside based on `http://api.openweathermap.org`.
   a. **app.js**:
      i.
```
function getCity()
```

II. User Manual

1. Convert temperature units displayed

- On Pebble: Double click the bottom button to tell the Arduino to switch whichever current temperature state it is in
- With sensor: Waive your hand in front of the motion sensor, the converted temperature will be displayed on both the Arduino as well as the watch. The watch will also show the time of the most recent motion.

2. Display local weather data

Press the select button to display local weather data, and set the light on the arduino based on the outside temperature compared to the inside temperature
Based on the temperature difference between the interior temperature:

- Green: Same as outside
- Red: It's hotter outside than inside
- Blue: It's colder outside than inside

3. Display most recent temperature sensor reading

For the past hour (if the sensor has been running for less than an hour, the user should be able to see the statistics for the time since the sensor started running).

Press the top button to retrieve the most recent temperature, as well as the average, high and low, and time of last motion.

4. Display the average, low, and high temperature

For the past hour (if the sensor has been running for less than an hour, the user should be able to see the statistics for the time since the sensor started running).

Press the top button to retrieve the most recent temperature, as well as the average, high and low, and time of last motion.

5. To switch only Pebble temperature unit display (i.e. Fahrenheit to Celsius; Celsius to Fahrenheit)

Double press the top button to change the UI display temperature

6. To switch only Arduino temperature unit 7-segment display (i.e. Fahrenheit to Celsius; Celsius to Fahrenheit)

Double press the bottom button to change the 7-segment display unit

7. Standby

- Enter: Press bottom Pebble button once
- Exit: Press bottom Pebble button once again

8. Terminating Middleware

- At any time, press 'q' to disconnect the server

9. Troubleshooting

- "No Device": middleware cannot get a reading from the sensor (e.g. because the sensor is disconnected)
- "What happened to your phone?": user interface becomes disconnected from the middleware (e.g. because of a network error or if the middleware stops running)
- "Timeout": user interface cannot reach API
- “Check your sensor” - sensor is disconnected

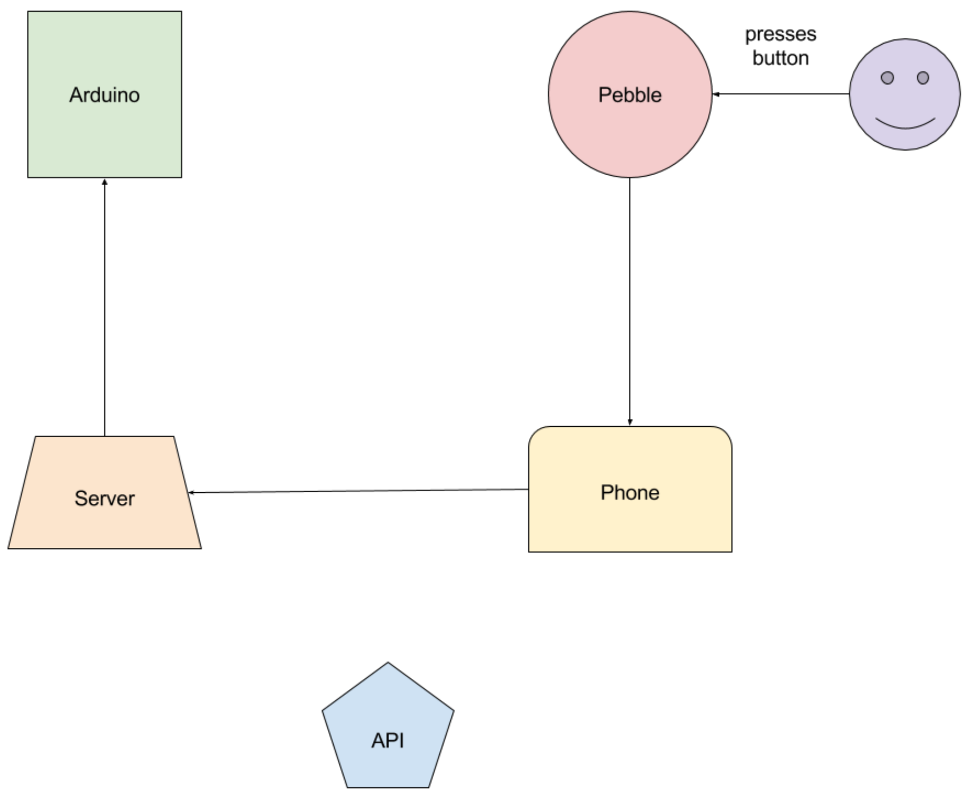
III. Supplemental Information

1. Where does data processing occur?

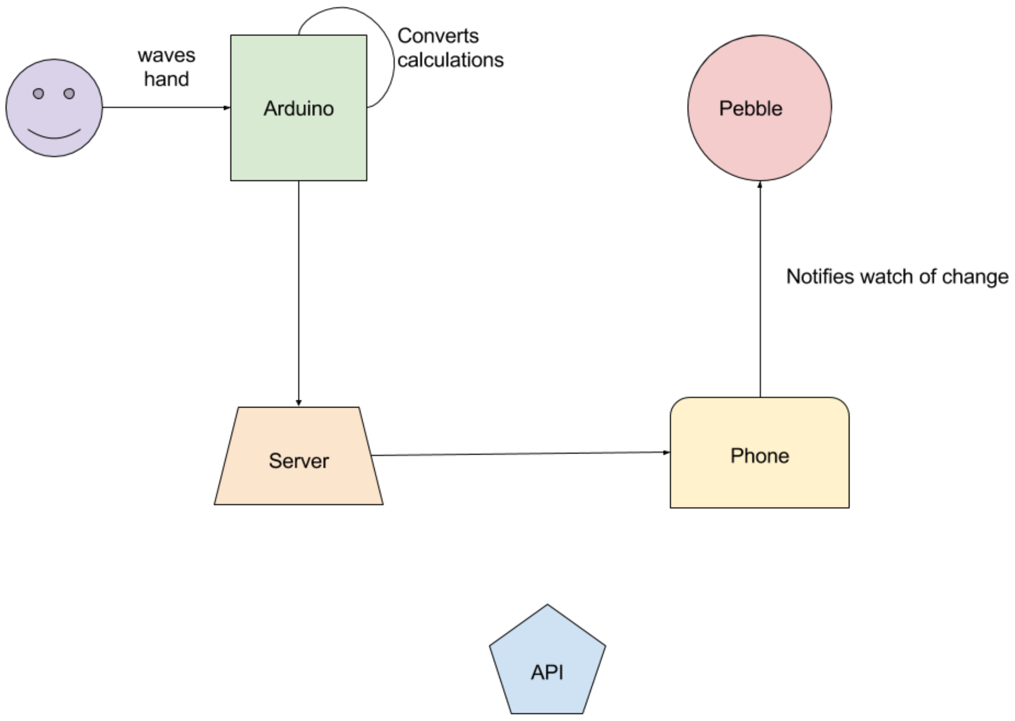
1. Celcius to Farenheit conversion: on Arduino
2. Interior/exterior temperature comparison - in the server
3. Show weather description - on the phone

2. What is the structure of your communication protocols between the different components?

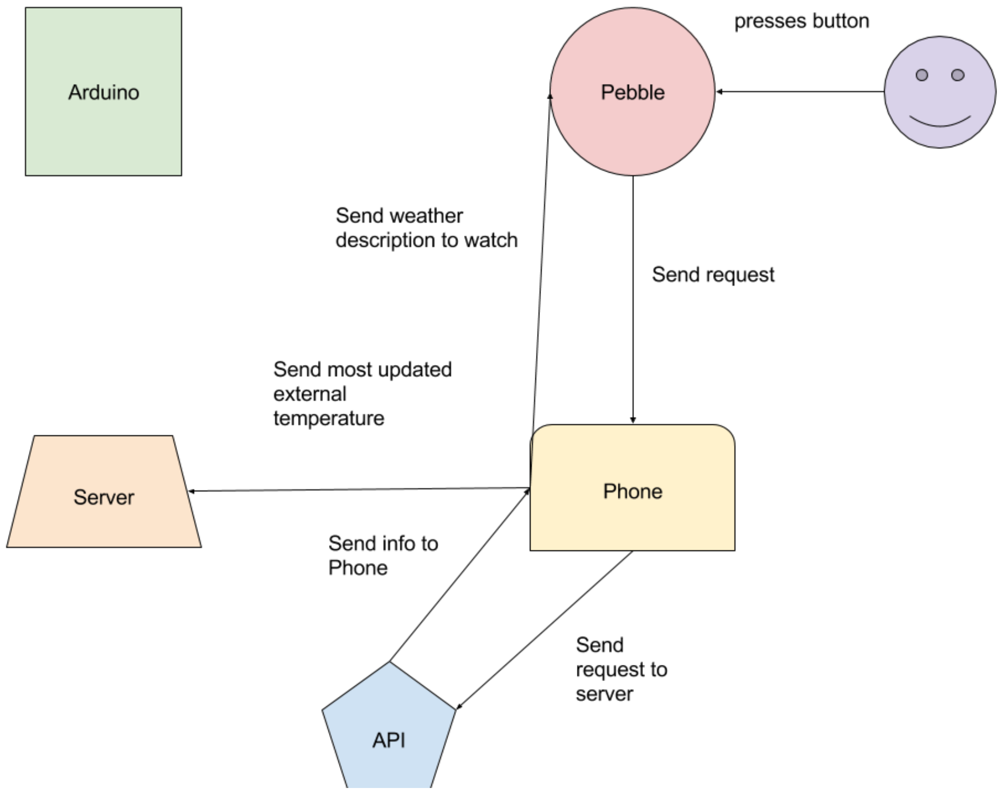
1. Additional Feature 1: Pebble button press changes color of Arduino light



2. Additional Feature 2: Motion sensor changes temperature units



3. Additional Feature 3: Show weather description (from API) on Pebble



3. How are API calls made?

- http://openweathermap.org/current
- https://developer.pebble.com/guides/communication/using-pebblekit-js/#using-geolocation

```
var key =
"http://api.openweathermap.org/data/2.5/forecast/city?id=524901&appid=863fad9850866cd53fbf3c264f6d4631"
```
[1]

JSON location for weather description:

```
var weather = JSONobj.weather[0].description; //cloudy, sunny, etc.
```

JSON location for Kelvin description:

```
var kel = JSONobj.main.temp;
```

[1] "&appid=..." denotes key